2009

Development Methods and a Scenegraph Animation API for Cluster Driven Immersive Applications

Brandon J. Newendorp
Iowa State University, brandon@newendorp.com

Christian J. Noon
Iowa State University, christian.noon@gmail.com

Chiu-Shui Chan
Iowa State University, cschan@iastate.edu

Eliot H. Winer
Iowa State University, ewiner@iastate.edu

James H. Oliver
Iowa State University, oliver@iastate.edu

DRAFT - 2009

Proceedings of the World Conference on Innovative VR 2009
Virtual Reality in Industrial Systems Design
WINVR09
February 25-26, 2009, Chalon-sur-Saône, France

WINVR09-711

A SCENEGRAPH ANIMATION API AND VR APPLICATION DESIGNED FOR LARGE-SCALE CLUSTERS

Brandon Newendorp[1]
Research Assistant
Iowa State University
Virtual Reality Application Center
Human Computer Interaction Program
Ames, Iowa, USA

Christian Noon
Research Assistant
Iowa State University
Virtual Reality Application Center
Dept. of Mechanical Engineering
Ames, Iowa, USA

Chiu-Shui Chan
Professor
Iowa State University
Virtual Reality Application Center
Dept. of Architecture
Ames, Iowa, USA

Eliot Winer
Assistant Professor
Iowa State University
Virtual Reality Application Center
Dept. of Mechanical Engineering
Ames, Iowa, USA

James H. Oliver
Professor
Iowa State University
Virtual Reality Application Center
Dept. of Mechanical Engineering
Ames, Iowa, USA

ABSTRACT

This paper presents the Virtual Universe—a virtual reality (VR) space exploration application designed to take advantage of a high-end VR system including: a high resolution six-walled display, an eight channel audio system, an IS-900 tracking system, and a 49 node cluster running dual graphics cards. The Virtual Universe is an immersive VR used to explore virtual space. In order to develop such an application to run efficiently on such a massive cluster, a scenegraph animation API—the Animation Engine—was constructed for software developers to smooth transitions and manipulations to scenegraph nodes. Inspired by Core Animation from Apple's Mac OS X Leopard, animation management no longer needs to take place on a frame-to-frame basis. The developer can use one line of code to enter the start state, end state, and number of frames to describe the animation, and the animation engine handles the rest in the background. Additional challenges such as maintaining high frame rates with 4096 x 4096 pixel textures, disabling redundant network traffic, and allowing the file server to leave requested data in cache were overcome. The architecture and development techniques for creating a stable immersive VR application on a high-end VR system is presented in this paper.

INTRODUCTION

Virtual reality (VR) is young technology under heavy academic and industrial development which can be traced back to as early as the 1960's [1]. Once computing and projection power advanced over the next two decades, academic institutions and industrial centers began investing in VR research and development. Before projection based displays were feasible, head-mounted displays were a common form of VR display system. Then, in the early 1990's, VR systems ranging from single-wall projection screens to four-walled CAVE displays [2] were developed across the world and have since become a very popular research area from both a hardware and software perspective. VR hardware has undergone many technological advances since its release in the early 1990's including projectors, tracking systems, interactive devices, and auditory interfaces.

The early 1990's projectors were capable of producing 1280 x 1024 pixel images on 10' x 10' display screens with a pixel resolution of approximately 91 pixels per square inch [3]. High-end projectors on the market today can produce up to 4096 x 2048 pixel images totaling over eight million pixels. When these projectors push images onto a 10' x 5' area, a pixel resolution of approximately 1165 pixels per square inch is created. This 1100% increase in pixel resolution gives the user

[1] Author of correspondence, Phone: (319) 269-7459, Email: bnew@iastate.edu

a much more clear and detailed display of the virtual environment than with the previous generations of VR systems.

More powerful projection system technology led to the development of interactive immersive environments. In order to interact with these immersive environments, tracking systems and interaction devices were developed. Early tracking systems used electromagnetic fields to perform location tracking [4]. Since these electromagnetic tracking systems could only achieve high accuracy in small environments, companies began investing money into researching new technological possibilities for tracking systems. For example, InterSense developed a tracking system combining ultrasound and inertial technology to track multiple devices simultaneously in a large-scale environment with high accuracy called the IS-900 [5]. The MiniTrax head tracker and wand units for the IS-900 can be seen below in Figure 1.

Figure 1: An InterSense IS-900 Wireless MiniTrax Head Tracker with a belt-worn batter/transmitter module (Left). An InterSense IS-900 Wireless MiniTrax Tracked Wand and single channel receiver (Right).

Auditory interfaces aid in enabling users to receive real-time feedback from virtual world applications. By combining both visual and auditory feedback within an immersive VR application, users typically feel more comfortable inside the environment. Additionally, three-dimensional (3D) sound can contribute to an even better sense of immersion within a 3D environment [6]. Visual VR environment events can have a much higher acceptance rate when accompanied by appropriate sounds emitted from their proper locations. 3D sound can also create situational awareness for events that take place out of the field of view of the user. In order to accomplish this, several VR systems have implemented 3D positional audio systems into the immersive environments.

BACKGROUND

With all these advanced technologies pooled together to form a very complex piece of hardware, the key task of VR development lies with writing the software to drive these technologies. 3D virtual world modeling programs such as i4D [7], Autodesk Maya [8] and Pixar's RenderMan [9] allow designers to create complex models and animations for virtual world scenes. Once these models are completed and exported into scenegraph friendly file formats, software engineers develop applications around these models by creating interactions internally between the models, or by implementing interaction devices to manipulate the objects within the virtual world.

A large gap exists however when a software developer needs to animate smooth transitions between these virtual models. No application programming interface (API) exists for performing smooth transitions and manipulations to scenegraph nodes such as objects, cameras, lights, etc. Instead, software developers write many complex functions to manipulate these object transformation matrices and geometric properties. The software framework i4D was developed to attempt to eliminate this problem by developing predefined animations for objects which they referred to as actors. Each type of actor had predefined actions which were built inside the i4D GUI. Once the actions were exported, an i4D VR application could be built using actors and their actions. However, objects could only be manipulated if they were predefined as an actor type. Additionally, animations are limited to actions defined by the actor type. To summarize, the i4D package is a scene designer tool, not a development tool. Currently, development tools for transitional scenegraph node animations do not exist.

The Virtual Universe is a VR application designed to take advantage of technologies in the C6. The C6 is an immersive VR system located in the Virtual Reality Applications Center [10] at Iowa State University and can seen below in Figure 2. This VR system features six rear-projected walls, each utilizing four Sony SXRD projectors per wall, [11] producing a 4096 x 4096 pixel image on each wall. An eight channel audio system and an InterSense IS-900 tracking system provides both the capability to position sounds and track a user's head position anywhere in the C6's 3D space. This world class VR system is paired with a Linux computer cluster containing a master node and 48 render nodes with dual NVIDIA Quadro FX 4500 graphics cards running in SLI.

Figure 2: Inside view of the C6 entrance (Right). A rack of Sony SXRD projectors for a single wall (Left).

Celestia [12] is an open source space simulation used as the motivation for creating the Virtual Universe. Celestia allows users to explore space across a huge range of scales from gigantic galaxies down to small spacecraft. The Virtual Universe is a immersive VR space exploration application, similar to Celestia. However, Celestia can only run on a desktop or laptop computer. The Virtual Universe allows users to navigate through virtual space consisting of several different systems and galaxies including the solar system, asteroid fields, nebulas, binary star systems, etc.

The Virtual Universe application employs VR Juggler [13, 14] and OpenSceneGraph (OSG) [15] as its VR and scenegraph platforms. VR Juggler provides necessary tools for VR application development such as interaction device communication, stereoscopic viewing, and display-device abstraction. OSG is an OpenGL-based, 3D graphics toolkit that provides real-time manipulation of a graphical scenegraph.

Autodesk 3ds Max [16] was used to create the 3D models in the Virtual Universe. The solar system planet models were generated using 4096x4096 pixel planet textures in order to push the cluster graphics cards to maximum capacity as seen in

Figure 3: The Virtual Universe running in C6 with the asteroid field and nebulas. (Left). The Virtual Universe running in C6 with Saturn and the sun in the background. (Right).

Figure 3. In order to generate smooth transitions between different systems of the Virtual Universe, the Animation Engine was developed. Inspired by Core Animation from Apple's Mac OS X Leopard [17], the developer no longer needs to manage individual transitions on a frame-to-frame basis. The developer could use one line of code to describe the animation, and the Animation Engine handles the rest in the background.

This paper presents an immersive VR application—the Virtual Universe—designed specifically to showcase the C6 technologies and VR concepts along with the Animation Engine, designed specifically manage scenegraph transitions with minimal developer effort. The following section of the paper presents a methodology section focusing on the architecture, Animation Engine, and hardware utilization of the Virtual Universe. Next is a development methods section describing the necessary steps taken to develop applications for large-scale computer clusters. Then, conclusions and future work sections are presented.

METHODOLOGY

System Architecture

The Virtual Universe code structure is wrapped around the idea of first making all the necessary updates to the scenegraph, then rendering and displaying the updates inside the current frame. For this reason, all calculations and updates need to be highly optimized to maintain an acceptable frame rate. If an application's frame rate drops below 30 frames per second (fps), users no longer see smooth animation within the scene. Therefore, the Virtual Universe is optimized to maintain at least 30 fps. Every frame, a number of steps occur sequentially in order to prepare the current frame for rendering and can be seen in the architecture schematic in Figure 4.

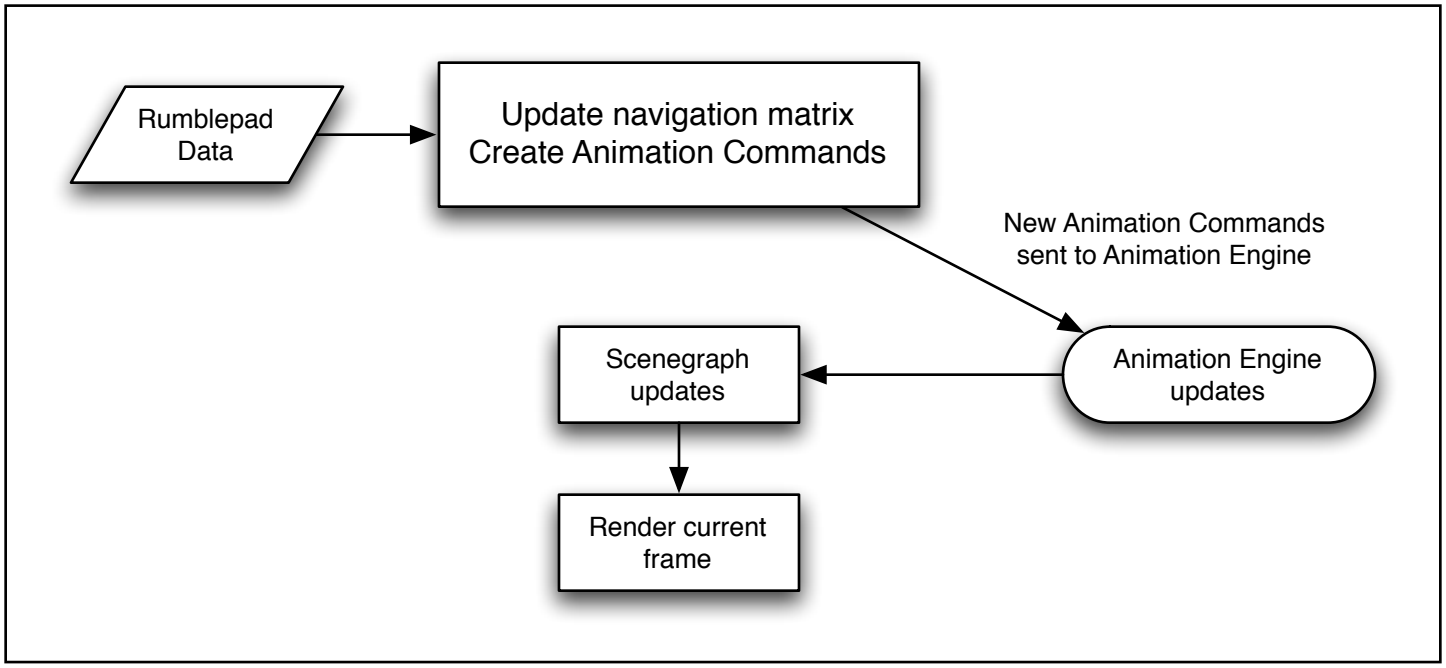

Figure 4: A schematic demonstrating the steps taken to render each frame of the Virtual Universe.

First, the master node in the rendering cluster receives updated data from the input device, a Logitech Cordless Rumblepad™ 2. The Rumblepad is the control device for the Virtual Universe, providing the application with controls for navigation, system transitions, and music playback. Data from the pair of analog joysticks on the Rumblepad is used to update the global navigation matrix, which is responsible for the user's head position and orientation within the virtual environment. The digital buttons on the Rumblepad are used to control the background music and transition between star systems in the Virtual Universe.

Once the master node has finished processing all the possible input commands, this information is sent to the cluster render nodes via VR Juggler's shared memory system, allowing every node to process input commands simultaneously. When a command to transition between star systems is received, each render node creates a new pair of Animation Commands. The first Animation Command instructs the Animation Engine to bring the requested system into view by bringing the system opacity to 100%. The second Animation Command instructs the Animation Engine to drop the opacity of the current system from 100% to 0% opacity. Both animations are performed for the same number of frames. The scenegraph is then allowed to prepare its final updates before rendering the next frame.

Animation Engine

The Animation Engine is a state-based framework, designed to enable developers to create animated visual effects with minimal coding effort. Without a system like the Animation Engine, a developer needs to write new code every time they want to change the state of an object in a scenegraph. This can result in duplicated code and is more susceptible to errors. Developers would also need to write code to interpolate between the current and destination states of an object. Commands for the Animation Engine are defined implicitly, where developers only need to specify the end state of the target. The Animation Engine handles interpolation from the current state to the end state.

In order to utilize the Animation Engine, a developer needs to create an Animation Command for every change they want to make to the scenegraph. An Animation Command has four key properties that need to be set: what property is changing (i.e. transparency, translation or scale), how long the animation should take, the end state of the animation, and the name of the scenegraph node to modify. For example, a new Animation Command could set the transparency of "Asteroid Root" to 0.5 in 60 frames.

Currently, there are three command types supported by the Animation Engine: transparency, translation and scale. A transparency command can change the opacity of a scenegraph node and its children to a given value. Translation commands are used to move the location of a scenegraph node and its children by providing the destination x, y, z coordinates of the node. Scale commands will resize a scenegraph node and its children by the provided scale value.

Once the developer has created a new Animation Command, they add the command to their application's singleton instance of the Animation Engine. At this point, the developer is no longer responsible for managing the animation. Every frame, the Animation Engine iterates through its internal list of active commands and makes necessary updates towards their end state by interpolating the changes from the node's original state to the end state. These calculations are performed every frame of the animation, not just when the command is created.

The Animation Engine is capable of processing multiple commands simultaneously. The maximum number of concurrent commands is only limited by the complexity of the scenegraph and the processing power of the computer system. If the Animation Engine is given a new command that supersedes a currently active command, it will automatically replace the prior command with the new one. This allows the Animation Engine to smoothly begin transitioning towards the new end state, rather than being required to finish outdated commands. For example, if an animation is underway to move Object A from 0, 0, 0 to 100, 100, 100. When Object A is at 50, 50, 50 (halfway to the end state), a new command is received to move Object A to 25, 25, 25. Rather than continuing to move to 100, 100, 100 before starting the new command, the Animation Engine will immediate begin moving Object A to 25, 25, 25 with the newly specified animation length.

Hardware Utilization

The C6 facility offers a wide range of features to improve user immersion in a virtual world. The Virtual Universe was conceived, designed, and developed to take advantage of nearly every feature the C6 has to offer. This contrasts with many other applications that, while capable of running in the C6, are designed to present all of their features on any VR display system instead of being designed specifically for the C6.

With the C6 utilizing Sony SXRD projectors and displaying a 4096 x 4096 wall resolution, the Virtual Universe was designed with extremely detailed models and textures to maintain high-fidelity visuals even at very close range. The Virtual Universe also takes advantage of this high resolution to increase the drawing distance in celestial scenes. This enables users to easily locate and identify distant objects because they still appear clear and visible thanks to the high resolution of both the displays and 3D models.

The C6 is a six walled CAVE, one of very few in the world, which allows users to be completely immersed in a virtual world. Although every scene in the Virtual Universe is displayed on all the walls in the C6, the asteroid field was created specifically for this capability. Users are able to fly through the asteroid field, watching asteroids fly past them on all six walls. The C6 also features a rear-projected floor surface, so multiple users can stand inside C6 without casting any shadows. This greatly improves the sense of presence in a virtual environment.

An eight channel audio system is installed in the C6, which provides the capability to position sounds anywhere in 3D space. The Virtual Universe utilizes the sound system to provide a different background music track for every star system users visit. These music tracks often are related to objects users see inside each system.

A final feature the Virtual Universe was designed to take advantage of is the InterSense IS-900 tracking system to track the main user's head position and orientation at all times. This provides two primary benefits to the Virtual Universe. First, the images on each wall are oriented to the main user's viewpoint, making the seams between the walls disappear. Second, the main user can walk around inside the C6, changing their point of view of the environment. This allows them to explore small objects in the Virtual Universe and examine them from various

perspectives. For example, the tracking system allows a user to explore inside the cabin of a space ship model or look around an asteroid simply by moving their head or walking around inside the C6.

DEVELOPMENT TECHNIQUES

In order to provide users with a fluid, visually appealing experience while using the Virtual Universe, a number of obstacles needed to be overcome in development. Although some of these obstacles are common to most VR applications developed today, several techniques were improved while creating the Virtual Universe. Special care was taken when designing the Virtual Universe to ensure it would work well on such a large computer cluster. Also, significant research was necessary to improve the performance of the textures used in the Virtual Universe. These techniques are described in this section.

Textures

During the development of the Virtual Universe, a key improvement was made to the textures in the application. All of the textures used in the model of our own solar system are 4096 x 4096 pixels while other textures range from 1024 x 1024 to 4096 x 4096 pixels. This is the highest resolution supported by the graphics cards in the C6 computer cluster—NVIDIA Quadro FX 4500s. In previous generations of the Virtual Universe, the textures were stored in the JPG file format because it offers high compatibility and effective compression.

However, a significant delay was observed when any computer in the render cluster encountered a new texture. When the application was first launched, users needed to make sure that every node of the computer cluster encountered every texture in the Virtual Universe before they could smoothly navigate through the application. This is because every time a texture was first seen by a graphics card, it had to load the texture from the file server into graphics memory. Despite having a gigabit Ethernet connection to the file server, this process still was slow enough that users could perceive a delay while textures load.

Through extensive research and investigation, it was discovered that the graphics cards are not efficient at loading standard JPG images as textures. There are a couple of reasons that alternative file formats, such as the DDS file format, are better suited to large numbers of high resolution textures. The DDS file format is the native file format for NVIDIA graphics cards. Two techniques were used to drastically improve texture loading time in the Virtual Universe. First, the graphics cards are required to add padding values to textures that are not square and in a power of 2. For example, data padding values would be added to a 813 x 642 texture, while a 1024 x 1024 texture requires no padding. All textures in the Virtual Universe were manually resized to eliminate any need for data padding. Also, NVIDIA graphics cards use a native pixel layout of BGRA. If a texture doesn't have a pixel layout of BGRA, the graphics card drivers are required to swizzle each texture as it is loaded. Swizzling is the process of rearranging the elements of a matrix, which is time consuming for the graphics card to perform. The DDS file format uses the BGRA pixel layout, eliminating the need for swizzling. Converting the textures in the Virtual Universe to the DDS file format and resizing the textures mitigate the previous generation problems with loading textures on each graphics card. According to tests run by NVIDIA, Quadro FX 4500 graphics cards can load and swizzle textures at a rate of 490 MB/s. When swizzling is not necessary, they are capable of loading textures at 2605 MB/s, a 5.3x increase in loading capabilities. Because the Virtual Universe uses 85MB of textures which need to be loaded on each of the 98 graphics cards, this is a significant difference in the time required to load all of the textures. [18]

Cluster Ready

Developing applications that run efficiently on a large cluster takes extra planning and development. The Virtual Universe was written and designed with these requirements in mind. Specifically, three techniques were used to improve the performance of the Virtual Universe on a cluster.

Because only one computer in the cluster is responsible for the audio playback, there's no need for the remainder of the cluster to receive audio commands or playback audio. The Virtual Universe application is designed so only one computer is playing back audio, allowing other computers to dedicate more processing power to visualization. Additionally, the Virtual Universe application instructs all nodes to load the models simultaneously at launch. This allows the file server to cache these files in RAM, drastically decreasing load times to about 5 minutes. Without this capability, nodes would have to load files sequentially, taking over 20 minutes to load the models.

Because network traffic can easily slow down a large cluster, care must be taken to minimize the inter-cluster network traffic while the application is running. Two steps were taken to eliminate extraneous network traffic: only sending necessary input data and disabling swap lock. When the master node gets input from the Rumblepad, it only passes that data on to the nodes if it is relevant. For example, navigation updates are sent over the network to nodes, but audio controls are not. The Virtual Universe has also disabled software swap lock, which is provided by VR Juggler. Swap lock is a software solution to ensure that the OpenGL render buffers are swapped simultaneously, which keeps all nodes rendering new images to the screens at the same time. Because it is a software solution, it generates dozens of network messages every frame. The C6 uses hardware frame locking, making software swap lock redundant. Eliminating the network traffic from swap lock also improved the performance of the Virtual Universe.

CONCLUSIONS

This paper presents the Virtual Universe—a virtual reality (VR) space exploration application designed to take advantage of a high-end VR system including: a high resolution six-walled display, an eight channel audio system, an IS-900 tracking system, and a 49 node cluster running dual graphics cards. The Virtual Universe is an immersive VR application used to explore space. In order to develop such an application to run efficiently on such a massive cluster, many techniques for improving performance were adopted in addition to the construction of a scenegraph animation API.

The scenegraph animation API—the Animation Engine—was constructed for software developers to smooth transitions and manipulations to scenegraph nodes. Inspired by Core Animation from Apple's Mac OS X Leopard, animation management no longer needs to take place on a frame-to-frame basis. The developer can use one line of code to describe the animation, and the Animation Engine handles the rest in the

background. Additional challenges also arose throughout the duration of the design process in order to design the Virtual Universe to adequately take advantage of the available VR technologies.

The first challenge was with creating high resolution models and images to display. In order to the highest resolution textures possible, specific file formates were necessary to provide better graphical performance. Other improvements were made to improve cluster performance by disabling audio playback on render nodes, eliminating redundant network traffic, and loading models simultaneously.

FUTURE WORK

Several additional features will be added into the Virtual Universe in the future. For the Animation Engine, support for a rotation scenegraph node manipulation will be developed. Also, the Animation Engine currently operates on a frame-based system which can cause issues when running on systems with higher/lower frame rates. To account for this, the Animation Engine will be converted to a time-based system which will be completely independent of frame rate. Another improvement will be to implement 3D positional audio to take advantage of the C6 eight channel audio system.

One final improvement to the visual aspect of the Virtual Universe will be to create more realistic space lighting. Currently, lighting in the Virtual Universe is done through OSG scene lighting. Space however is lit from the light emitted from stars and is then reflected off celestial bodies. To replicate this effect, OSG lights must be embedded within the stars and suns of the different systems. This will give the end user a much more realistic feel for accurate lighting and shadowing that occurs in space.

REFERENCES

[1] Sutherland, I. E., *The Ultimate Display*, Proceedings of IFIPS Congress, New York City, NY, May 1965, Vol. 2, pp. 506-508.

[2] Cruz-Neira, Carolina, Sandin, Daniel J., and DeFanti, Thomas A., "Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE," *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, September 1993, pp. 135-142.

[3] Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., Hart, J. C., "The CAVE: Audio Visual Experience Automatic Virtual Environment," *Communications of the ACM*, ACM, June 1992, Vol. 35, No. 6, pp. 64-72.

[4] Kindratenko, V., "A Comparison of the Accuracy of an Electromagnetic and a Hybrid Ultrasound-inertia Position Tracking System," *Presence: Teleoperators and Virtual Environments*, MIT Press, Cambridge, MA, December 2001, Vol. 10, Issue 6, pp. 657-663.

[5] Wormell, D. and Foxlin, E., "Advancements in 3D Interactive Devices for Virtual Environments," *Proceedings of the Workshop on Virtual Environments 2003*, ACM, Zurich, Switzerland, 2003, Vol. 39, pp. 47-56.

[6] Begault, D. R., *3D Sound for Virtual Reality and Multimedia*, Academic Press Profession, Inc., San Diego, CA, 1994, ISBN: 0120847353.

[7] Geiger, C., Paelke, V., Reimann, C., and Rosenback, W., "A Framework for the Structured Design of VR/AR Content," *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, ACM, Seoul, Korea, 2000, pp. 75-82.

[8] Pixar's RenderMan: https://renderman.pixar.com/, accessed June 2008.

[9] Autodesk Maya: http://usa.autodesk.com/adsk/servlet/index?id=7635018&siteID=123112, accessed June 2008.

[10] Virtual Reality Applications Center: http://www.vrac.iastate.edu/, accessed June 2008.

[11] Sony SRX-S110 Projector: http://pro.sony.com/bbsc/ssr/product-SRXS110/, accessed June 2008.

[12] Celestia: http://www.shatters.net/celestia/, accessed June 2008.

[13] VR Juggler, http://www.vrjuggler.org, accessed January 2008.

[14] Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., Cruz-Neira, C., "VR Juggler: A Virtual Platform for Virtual Reality Application Development," *IEEE Virtual Reality Conference 2001 (VR 2001)*, IEEE, Yokohama, Japan, March 13-17, 2001, pp. 89-96.

[15] OpenSceneGraph, http://www.openscenegraph.org, accessed January 2008.

[16] Autodesk 3DS MAX, http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=5659302, accessed January 2008.

[17] Mac OS X Leopard Core Animation: http://www.apple.com/macosx/technology/coreanimation.html, accessed June 2008.

[18] NVIDIA Technical Brief: http://http.download.nvidia.com/developer/Papers/2005/Fast_Texture_Transfers/Fast_Texture_Transfers.pdf, accessed June 2008.